Plateforme matérielle–logicielle à bas coût pour l'émulation de fautes

Pierre Guilloux et Arnaud Tisserand. IRISA – CNRS – UR1

ARDyT

Reliasic

1. Objectif : circuits arithmétiques robustes aux fautes

Étude des liens entre les techniques de détection/tolérance et

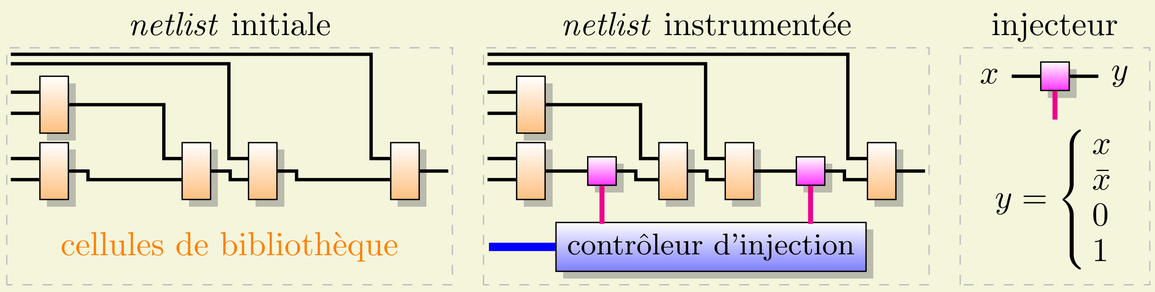
- types d'opérations $\pm, \times, \times_{\text{cst}}, \div, \sqrt{}, \sin, \cos, \log, (\pm, \times)\text{mod}, \ldots$
- algorithmes de calcul
- représentations des nombres
- diverses optimisations aux niveaux architecture & circuit
- niveau d'erreur mathématique due aux fautes et arrondis

Modèles précis ⇒ simulations intensives

Applications : traitement du signal et des images, calcul embarqué, cryptographie asymétrique

2. Technique: émulation de fautes sur FPGA

Utilisée depuis 30 ans, elle consiste à implanter sur FPGA une version instrumentée de la *netlist* pour simuler des fautes

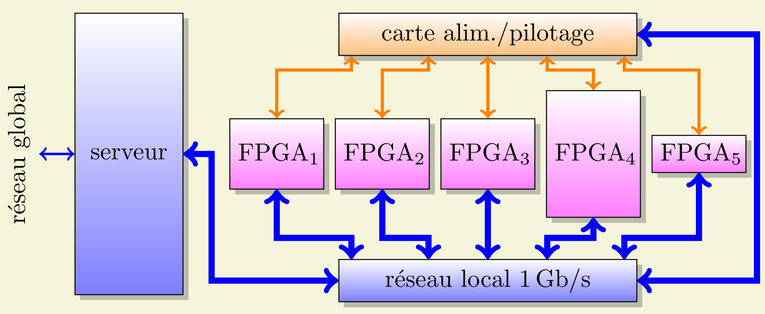


3. Description de la plateforme

Petit cluster de cartes FPGA et de serveur(s) pour les outils CAO

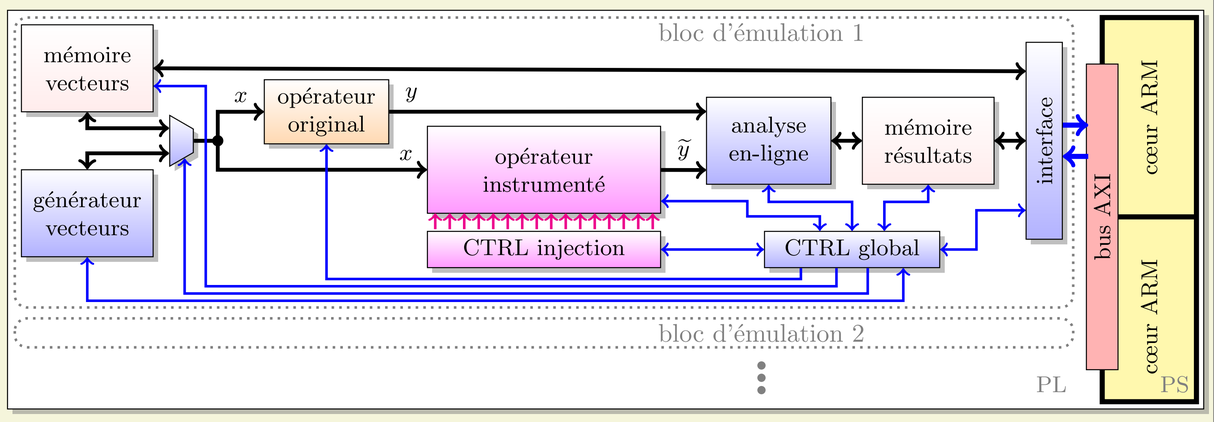
1	serveur 8 cœurs 4 GHz
3	FPGA Zedbord 7020
1	FPGA PicoZed 7035
1	FPGA Zybo 7010
1	carte alim. / pilotage
1	*switch* ethernet 1Gb/s

Coût total 3500 €HT



Extension facile à des dizaines de cartes et quelques serveurs

4. Blocs d'émulation matérielle dans un des FPGA



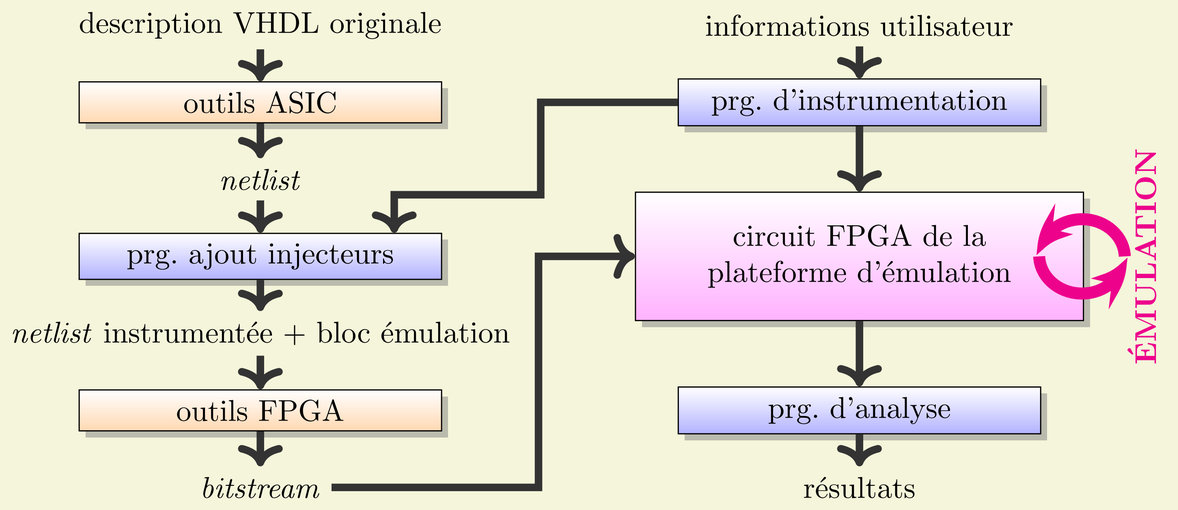
- opérateur original (référence) calcule $y = f(x)$
- opérateur instrumenté calcule $\widetilde{y}$ (sortie « fautée »)
- contrôleur d'injection pilote les scénarios définis
- mémoire et générateur de vecteurs fournissent des entrées x
- unité d'analyse en-ligne et mémoire temporaire
- contrôle global du bloc d'émulation
- interface vers le Linux embarqué PS (et le serveur)

Références:

Bibliothèque opérateurs arithmétiques R. Zimmermann (http://www.iis.ee.ethz.ch/ zimmi/arith_lib)

P. Guilloux & A. Tisserand, papier conférence Compas 2016

5. Flot logiciel



Répartition des points d'injection (actuellement) :

- aléatoire sur les signaux internes
- aléatoire uniforme sur la surface
- aléatoire avec prise en compte des délais (travail futur)

6. Exemple: multiplieur protégé par code résidu

Multiplieur entier 16×16 bits et moduli $m \in \{3, 7, 15\}$ avec

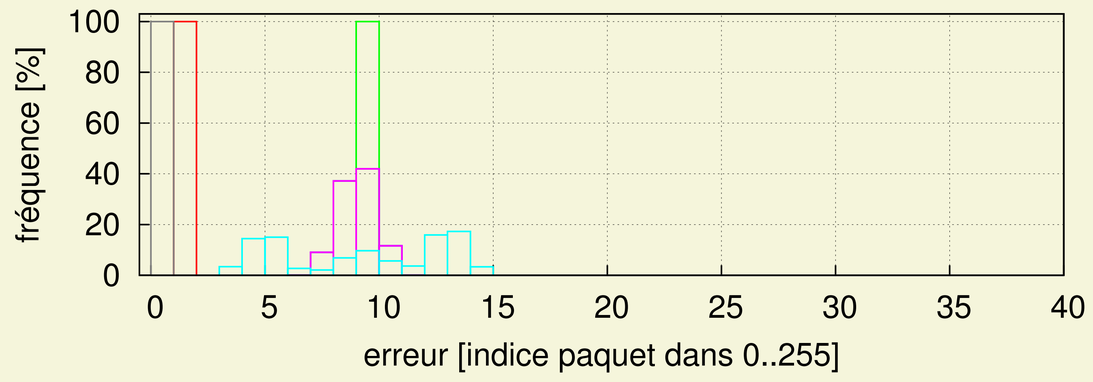
$$(a \times b) \bmod m = \big((a \bmod m) \times (b \bmod m)\big) \bmod m$$

étape et outil	durée	surface	délai
Impl. ASIC mult. original	25 s	1164 portes	–
Impl. FPGA mult. original	43 s	444 LUT	9.4 ns
Impl. FPGA mult. instr.	48 s	453 LUT	10.1 ns
Impl. FPGA bloc émulation	389 s	1714 LUT + 1 BRAM	11.0 ns
boot petalinux Zedboard	15 s	–	–
émulation 1 faute & 1 bloc	94.5 s	–	–

Comparaison à une simulation CABA (Vivado 2014.4) :

accélération ×6880 sur 1 bloc d'émulation

Test exhaustif (2^{32} entrées) de la distribution d'erreur pour différentes erreurs simples :



Étude des coûts/performances/taux de détection :

modulo m	surface [LUT]	délai [ns]	taux détection moyen mesuré [%]
3	525	10.4	66.7
7	526	10.1	85.6
15	553	13.0	93.3

7. Travaux en cours et futurs

Déployer un système de gestion et d'ordonnancement pour grappe de machines (SLURM, `http://slurm.schedmd.com/`)

Performances attendues de la plateforme :

100 blocs à 100 MHz	/ s	/ h	/ j
nb. fautes ou entrées	$100 \cdot 10^6 \times 100 \approx 2^{33}$	$\approx 2^{45}$	$\approx 2^{49}$

ANR-11-INSE-15 http://ardyt.irisa.fr/ & Labex CominLabs http://www.reliasic.cominlabs.ueb.eu/

ANR – Projet financé par l'ANR – Project funded by the ANR

CominLabs

UMR IRISA

cnrs

Université de Rennes 1